Network scanning, Anonymization Networks

CS 1660: Introduction to Computer Systems Security
Ports, Scanning, and Firewalls
How to support multiple applications?

Network layer: moving data between hosts
Transport layer: Abstraction for getting data data to different applications on a host
The Transport Layer

Network layer: moving data between hosts
Transport layer: Abstraction for getting data data to different applications on a host

- Multiplexing multiple connections at same IP with port numbers
- Series of packets => stream of data/messages
- May provide: reliable data delivery

Two key protocols: TCP, UDP
From earlier: OSI Model

One or more nodes within the network

End host

Application
Presentation
Session
Transport
Network
Data link
Physical

Application Protocol

End host

Application
Presentation
Session
Transport
Network
Data link
Physical

Transport Protocol

Network Protocol

Link-Layer Protocol

One or more nodes within the network
What’s a port number?

• 16-bit unsigned number, 0-65535
• Ports define a communication *endpoint*, usually a process/service on a host
• OS keeps track of which ports map to which applications
What’s a port number?

• 16-bit unsigned number, 0-65535
• Ports define a communication *endpoint*, usually a process/service on a host
• OS keeps track of which ports map to which applications

*Port numbering*

• port < 1024: “Well known port numbers”
• port >= 20000: “ephemeral ports”, for general app. use
Some common ports

<table>
<thead>
<tr>
<th>Port</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>20, 21</td>
<td>File Transfer Protocol (FTP)</td>
</tr>
<tr>
<td>22</td>
<td>Secure Shell (SSH)</td>
</tr>
<tr>
<td>23</td>
<td>Telnet (pre-SSH remote login)</td>
</tr>
<tr>
<td>25</td>
<td>SMTP (Email)</td>
</tr>
<tr>
<td>53</td>
<td>Domain Name System (DNS)</td>
</tr>
<tr>
<td>67, 68</td>
<td>DHCP</td>
</tr>
<tr>
<td>80</td>
<td>HTTP (Web traffic)</td>
</tr>
<tr>
<td>443</td>
<td>HTTPS (Secure HTTP over TLS)</td>
</tr>
</tbody>
</table>
How ports work

Two modes:

• Applications "listen on" or "bind to" a port to wait for new connections

• Hosts make connections to a particular IP and port
Two modes:

• Applications "listen on" or "bind to" a port to wait for new connections
  => Example: webserver listens on port 80

• Hosts make connections to a particular IP and port
  => Example: client connects to <webserver IP>, port 80
    (eg. 1.2.3.4:80)
A must know B is listening on port 80 => "well known numbers"!

When connecting, A's OS picks random source port (eg. 12345), used for its side of connection
connect(1.2.3.4, 80)

A
1.2.3.4

Src IP 1.2.3.4
Port 12345

Dst IP 5.6.7.8
Port 80

B
5.6.7.8

listen(80)

B responds to A using this port

Src IP 5.6.7.8
Port 80

Dst IP 1.2.3.4
Port 12345
Sockets

OS keeps track of which application uses which port

Two types:

• **Listening ports**
• **Connections** between two hosts (src/dst port)
OS keeps track of which application uses which port

**Two types:**

- **Listening ports**
- **Connections** between two hosts (src/dst port)

Socket: OS abstraction for a network connection, like a file descriptor

Socket table maps: port => socket
```
<table>
<thead>
<tr>
<th>Proto</th>
<th>Recv-Q</th>
<th>Send-Q</th>
<th>Local Address</th>
<th>Foreign Address</th>
<th>(state)</th>
</tr>
</thead>
<tbody>
<tr>
<td>tcp6</td>
<td>0</td>
<td>0</td>
<td><em>.</em>.22</td>
<td><em>.</em></td>
<td>LISTEN</td>
</tr>
<tr>
<td>tcp4</td>
<td>0</td>
<td>0</td>
<td><em>.</em>.51036</td>
<td><em>.</em></td>
<td>LISTEN</td>
</tr>
<tr>
<td>tcp4</td>
<td>0</td>
<td>0</td>
<td>127.0.0.1.9999</td>
<td><em>.</em></td>
<td>LISTEN</td>
</tr>
<tr>
<td>tcp4</td>
<td>0</td>
<td>0</td>
<td>10.3.146.161.51094</td>
<td>104.16.248.249.443</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp4</td>
<td>0</td>
<td>0</td>
<td>10.3.146.161.51076</td>
<td>172.66.43.67.443</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp6</td>
<td>0</td>
<td>0</td>
<td>2620:6e:6000:900.51074</td>
<td>2606:4700:3108:443</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp4</td>
<td>0</td>
<td>0</td>
<td>10.3.146.161.51065</td>
<td>35.82.230.35.443</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp4</td>
<td>0</td>
<td>0</td>
<td>10.3.146.161.51055</td>
<td>162.159.136.234.443</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp4</td>
<td>0</td>
<td>0</td>
<td>10.3.146.161.51038</td>
<td>17.57.147.5.5223</td>
<td>ESTABLISHED</td>
</tr>
</tbody>
</table>
```

netstat -an: Show all connections
netstat -lnp: Show listening ports + applications using them (as root)
Transport protocols: TCP, UDP, ...

Transport protocol => *how* application exchanges data

- **UDP**: small, discrete messages
  - Used by: DNS, DHCP, Custom protocols
Transport protocols: TCP, UDP, ...

Transport protocol => how application exchanges data

- **UDP**: small, discrete messages
  - Used by: DNS, DHCP, Custom protocols

- **TCP**: app sends stream of bytes, OS divides into packets and figures out how to send reliably
  - Used by: HTTP(S), SSH
  - Connections have "state" => Extra info in packet headers, OS state
Transport protocols: TCP, UDP, ...

Transport protocol => *how* application exchanges data

- **UDP**: small, discrete messages
  - Used by: DNS, DHCP, Custom protocols

- **TCP**: app sends stream of bytes, OS divides into packets and figures out how to send reliably
  - Used by: HTTP(S), SSH
  - Connections have "state" => Extra info in packet headers, OS state

=> Most applications build on one of these, defines what packets look like on the wire
Transport protocols: TCP, UDP, ...

Transport protocol => how application exchanges data

• UDP: small, discrete messages
  • Used by: DNS, DHCP, Custom protocols

• TCP: app sends stream of bytes, OS divides into packets and figures out how to send reliably
  • Used by: HTTP(S), SSH
  • Connections have "state" => Extra info in packet headers, OS state

=> Most applications build on one of these, defines what packets look like on the wire

Want to know a LOT more? Take CS1680!
Transport protocols: TCP, UDP, ...

Want to know more? Take CS1680!
Why do we care?

If a listening port is open, you can send data to an application
=> Defines attack surface on network!

Implications for:
• How to find vulnerable hosts/services
• How we protect them
Port scanning

What can we learn if we just start connecting to well-known ports?

• Applications have common port numbers
• Network protocols use well-defined patterns
Port scanning

What can we learn if we just start connecting to well-known ports?

• Applications have common port numbers
• Network protocols use well-defined patterns

Port scanners: try to connect to lots of ports, determine available services, find vulnerable services...
**nmap**

nmap: Widely-used network scanning tool

- Scan ranges of IPs, look for specific open ports
- Scan many ports on specific hosts, learn about available services
- Lots of extensions/scripts...

```
$ nmap -sV -A 172.17.48.44
Nmap scan report for 172.17.48.25
Host is up (0.00065s latency).
Not shown: 997 closed ports

<table>
<thead>
<tr>
<th>PORT</th>
<th>STATE</th>
<th>SERVICE</th>
<th>VERSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>22/tcp</td>
<td>open</td>
<td>ssh</td>
<td>OpenSSH 6.2 (protocol 2.0)</td>
</tr>
<tr>
<td>88/tcp</td>
<td>open</td>
<td>kerberos-sec</td>
<td>Heimdal Kerberos (server time: 2023-04-25 15:04:20Z)</td>
</tr>
<tr>
<td>5900/tcp</td>
<td>open</td>
<td>vnc</td>
<td>Apple remote desktop vnc</td>
</tr>
</tbody>
</table>

Service Info: OS: Mac OS X; CPE: cpe:/o:apple:mac_os_x
```
Different OSes use different defaults in packet headers
=> Can use for detection!

<table>
<thead>
<tr>
<th></th>
<th>linux 2.4</th>
<th>linux 2.6</th>
<th>openbsd</th>
<th>MACOS X</th>
<th>windows</th>
</tr>
</thead>
<tbody>
<tr>
<td>ttl</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>128</td>
</tr>
<tr>
<td>packet length</td>
<td>60</td>
<td>60</td>
<td>64</td>
<td>64</td>
<td>48</td>
</tr>
<tr>
<td>initial windows</td>
<td>5840</td>
<td>5840</td>
<td>16384</td>
<td>9000</td>
<td>16384</td>
</tr>
<tr>
<td>mss</td>
<td>512</td>
<td>512</td>
<td>1460</td>
<td>1460</td>
<td>1460</td>
</tr>
<tr>
<td>ip id</td>
<td>0</td>
<td>random</td>
<td>random</td>
<td>random</td>
<td>increment</td>
</tr>
<tr>
<td>enabled tcp opt</td>
<td>MNNTNW</td>
<td>MNNTNW</td>
<td>M</td>
<td>M</td>
<td>MNW</td>
</tr>
<tr>
<td>timestamp inc.</td>
<td>100hz</td>
<td>1000hz</td>
<td>unsupported</td>
<td>unsupported</td>
<td>100Hz</td>
</tr>
<tr>
<td>sack</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
</tr>
<tr>
<td>SYN attempts</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
Large-scale port scanning

Can reveal lots of open/insecure systems!

**Examples:**

- shodan.io
- VNC roulette
- Open webcam viewers...
- ...

Disclaimer

• Network scanning is often very easy to detect

• Unless you are the owner of the network, it’s seen as malicious activity

• If you scan the whole Internet, the whole Internet will get mad at you (unless done very politely)

• Do NOT try this on the Brown network. We warned you.
Scanning I have done

• Scanned IPv4 space for ROS (Robot Operating System)
• Found ~200 “things” using ROS (some robots, some other stuff)
How to defend in the network?
How to defend ports?

Firewall: set of policies to block/monitor access

=> Could be a single box, an OS feature, or a cloud-based service (think CDN)
How to defend ports?

Firewall: set of policies to block/monitor access

=> Could be a single box, an OS feature, or a cloud-based service (think CDN)
How to defend ports?

Firewall: set of policies to block/monitor access

• Simple: rules based on packet headers
• Expensive: look at packet contents like HTTP headers/data
  ⇒ Deep Packet Inspection (DPI)

• Linux: iptables/netfilter: firewall/filtering in the Linux kernel
Firewall policy example: stateless rules

```
[root@Warspite deemer]# iptables -L -n

Chain OUTPUT (policy ACCEPT)
    ....

Chain INPUT (policy ACCEPT)

    target prot opt source               destination
    DROP  tcp --  0.0.0.0/0               0.0.0.0/0               tcp dpt:80
    DROP  tcp --  0.0.0.0/0               0.0.0.0/0               tcp dpt:3389
    DROP  *      138.16.0.0/16           0.0.0.0/0
    DROP  *      138.16.0.0/16           0.0.0.0/0
    ACCEPT all --  0.0.0.0/0               0.0.0.0/0
```

Default: accept traffic except...
Firewall policy example: stateless rules

```
[root@Warsprite deemer]# iptables -L -n

Chain OUTPUT (policy ACCEPT)
....

Chain INPUT (policy ACCEPT)
target prot opt source          destination
DROP  tcp  --  0.0.0.0/0       0.0.0.0/0
DROP  tcp  --  0.0.0.0/0       0.0.0.0/0
DROP  *     --  138.16.0.0/16  0.0.0.0/0
DROP  *     --  138.16.0.0/16  0.0.0.0/0
ACCEPT all  --  0.0.0.0/0       0.0.0.0/0

```

Default: accept traffic except...

- Drop packets from specific hosts
- Drop packets arriving on specific ports

DoS, DNS, TLS 4/25/23
### Firewall policy example: stateful rules

**Default:** drop traffic except...

```
[root@Warsprite deemer]# iptables -L -n
```

<table>
<thead>
<tr>
<th>Chain</th>
<th>INPUT (policy DROP)</th>
<th>target</th>
<th>prot opt</th>
<th>source</th>
<th>destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCEPT</td>
<td>all -- 0.0.0.0/0 0.0.0.0/0</td>
<td>state RELATED,ESTABLISHED</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DROP</td>
<td>tcp -- 0.0.0.0/0 0.0.0.0/0</td>
<td>tcp dpt:22 state NEW recent: SET name: SSH side: source mask: 255.255.255.255</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACCEPT</td>
<td>tcp -- 0.0.0.0/0</td>
<td>udp -- 0.0.0.0/0 0.0.0.0/0 state NEW udp dpt:53 recent: UPDATE seconds: 60 hit_count: 15 name: LDNS side: source mask: 255.255.255.255</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DROP</td>
<td>udp -- 0.0.0.0/0 0.0.0.0/0</td>
<td>udp dpt:53 recent: UPDATE seconds:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACCEPT</td>
<td>tcp -- 0.0.0.0/0 0.0.0.0/0</td>
<td>state NEW tcp dpt:53</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACCEPT</td>
<td>udp -- 0.0.0.0/0 0.0.0.0/0</td>
<td>state NEW udp dpt:53</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACCEPT</td>
<td>tcp -- 0.0.0.0/0 0.0.0.0/0</td>
<td>state NEW tcp dpt:443</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Firewall policy example: stateful rules

Default: drop traffic except...

```
[root@Warsprite deemer]# iptables -L -n

Chain INPUT (policy DROP) target prot opt source destination
ACCEPT   all -- 0.0.0.0/0 0.0.0.0/0  state RELATED,ESTABLISHED
\           tcp -- 0.0.0.0/0 0.0.0.0/0
DROP     tcp -- 0.0.0.0/0 0.0.0.0/0  tcp dpt:22 state NEW recent: SET name: SSH side: source
\           tcp dpt:22 state NEW recent: UPDATE seconds: 60
\           hit_count: 8 T0.0.0.0/0  state NEW tcp dpt:22
ACCEPT   tcp -- 0.0.0.0/0 0.0.0.0/0  udp -- 0.0.0.0/0 0.0.0.0/0  state NEW udp dpt:53
\           udp dpt:53 recent: UPDATE seconds: hit_count: 15 name: LDNS side: source
DROP     udp -- 0.0.0.0/0 0.0.0.0/0  udp dpt:53 state NEW udp dpt:53
ACCEPT   tcp -- 0.0.0.0/0 0.0.0.0/0  state NEW tcp dpt:53
ACCEPT   udp -- 0.0.0.0/0 0.0.0.0/0  state NEW udp dpt:53
ACCEPT   tcp -- 0.0.0.0/0 0.0.0.0/0  state NEW tcp dpt:443
```

- Allow new connections only to certain ports
- Rate-limiting on high-traffic ports
After scanning: what else can you do?
After scanning: what else can you do?

Starting point for more attacks

• Scans may indicate unprotected services
• Fingerprinting info may show services vulnerable to known exploits

=> Automated tools to do this at scale (eg. Metasploit)
<table>
<thead>
<tr>
<th>IP Address</th>
<th>Name</th>
<th>OS Name</th>
<th>Version</th>
<th>Purpose</th>
<th>Services</th>
<th>Vulns</th>
<th>Notes</th>
<th>Updated</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1.55.113</td>
<td>vmware-bavm</td>
<td></td>
<td></td>
<td>device</td>
<td>1</td>
<td>1</td>
<td></td>
<td>3 minutes ago</td>
<td>Shelled</td>
</tr>
<tr>
<td>10.1.55.253</td>
<td>Konica Printer</td>
<td>printer</td>
<td></td>
<td>printer</td>
<td>1</td>
<td></td>
<td></td>
<td>5 minutes ago</td>
<td>Scanned</td>
</tr>
</tbody>
</table>

Showing 1 to 3 of 3 entries
Anonymization networks
Internet Censorship

- Control or suppression of the publishing or accessing of information on the Internet
- Carried out by governments or by private organizations either at the behest of government or on their own initiative
- Individuals and organizations may engage in self-censorship on their own or due to intimidation and fear.
- Comparitech Internet Censor map 2022
Filtering vs. Censoring

ContentBarrier

Your request has been blocked by ContentBarrier.

This web site has been blocked because it matches the following forbidden category: Sex/Pornography.

If you think this web site does not match this category, please submit this URL.

You can ask your parent or your administrator to unlock this web site (a password will be required).

Submit this link  Allow this web site...

The connection was reset

The connection to the server was reset while the page was loading.

- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the Web.

Try Again
Censoring Techniques

• **DNS blacklist**
  • DNS does not resolve domain names or returns incorrect IP addresses, e.g., www.google.com returns 'page not found’

• **IP blacklist**
  • For sites on a blacklist, the censoring system prevents connection attempts

• **Keyword blacklist**
  • The censoring system scans the URL string (e.g., search terms) and interrupts the connection if it contains keywords from a blacklist
OONI

• Open Observatory of Network Interference
• a project that monitors internet censorship globally
• https://ooni.org/
The Onion Router
Overview

• First the US Naval Research Laboratory, then the EFF and now the Tor Project (www.torproject.org)
• Access normal Internet sites anonymously, and Tor hidden services.
• Locally run SOCKS proxy that connects to the Tor network.
• “Tor is free software and an open network that helps you defend against a form of network surveillance that threatens personal freedom and privacy, confidential business activities and relationships, and state security known as traffic analysis.” [TOR project website]
Anonymity

• Preventing identification within a group
  • E.g., departmental VPN, home NAT router
  • Group should be as large as possible

• Preventing association of action and identity
  • E.g., distributed denial of service by hidden attacker
Mix

Trusted router, Rose
Public-key encryption
Message from Alice to Bob via Rose
\[ E_{KR}(Bob, E_{KB}(M)) \]

Precautions
- Fixed message size
- Continuous communication
- Dummy messages
- Chain of mixes
How Tor Works: 1

Step 1: Alice's Tor client obtains a list of Tor nodes from a directory server.
Onion Routing

Group of routers
Message sent via random sequence of routers
Layered encryption
  Build onion inside out
Routing
  Peel onion outside in
Each router knows previous and next
How Tor Works: 2

Step 2: Alice's Tor client picks a random path to destination server. Green links are encrypted, red links are in the clear.
Onion Routing in Practice

Do not encrypt final hop
   Encryption may be done by application (e.g., https)

Source sets up
   Random circuit (route)
   Symmetric keys shared with routers

Data tunneled to final router over circuit
Step 3: If at a later time, the user visits another site, Alice's Tor client selects a second random path. Again, green links are encrypted, red links are in the clear.
Types of relays on the Tor network

**Guard and Middle relay** (non-exit relays)
- Guard relay first relay in the chain of 3 relays building a Tor circuit
- Middle relay acts as an intermediate hop between the Guard and exit

**Exit relay**
- Final relay in a Tor circuit
- Eg: A website will see the *exit relay IP* instead of the *real IP address* of the Tor user
- Greatest legal exposure and liability of all the relays
DEMOS

- www.eff.org/pages/tor-and-https
- torproject.org
- Guard, middle, Exit nodes
- Exit nodes list
  - https://check.torproject.org/torbulkexitlist
Applications/Sites

- **Hidden services**
  Normally websites, but can be just about any TCP connection

- Tor Hidden Service Example (Hiddenwiki): http://zqktlwi4fecvo6ri.onion

- Duckduckgo.com - https://duckduckgogg42xjoc72x3sjasowoarfbgcmvfimaftt6twagswzcjad.onion/

- Facebook www.facebookcorewwwwi.onion/

- **.onion TLD v2:**
  - non-mnemonic,
  - 16-character alpha-semi-numeric hashes
  - automatically generated based on a public key when a hidden service is configured
  - “vanity address” possible with expensive computation
    https://blog.torproject.org/v2-deprecation-timeline/

Privacy & Censorship
TOR Analysis

Advantages
• Tunnel, through a SOCKS proxy, allows to work any protocol.
• Three nodes of proxying, each node not knowing the one before last, makes very difficult to find the source.

Problems
• Slow (high latency)
• Exit node?
• Semi-fixed Infrastructure: possible to block all Tor relays listed in the Directory. Bridged node.
• Fairly easy to tell someone is using it from the server side
Identify TOR traffic

Default configuration:

- **Local**
  - 9050/tcp Tor SOCKS proxy
  - 9051/tcp Tor control port
  - 8118/tcp Privoxy

- **Remote**
  - 443/tcp and 80/tcp mostly
  - Servers may also listen on port 9001/tcp, and directory information on 9030
Clicker Question (2)

How To Block Tor? Attackers can block users from connecting to the Tor network, in which way?

A. Blocking the directory authorities  
B. Blocking all the relay IP addresses in the directory  
C. Filtering based on Tor's network fingerprint  
D. Preventing users from finding the Tor software  
E. All the above
Clicker Question (2) - Answer

How To Block Tor? Attackers can block users from connecting to the Tor network, in which way?

A. Blocking the directory authorities
B. Blocking all the relay IP addresses in the directory
C. Filtering based on Tor's network fingerprint
D. Preventing users from finding the Tor software
E. All the above
Bridge relays

• Rather than signing up as a normal relay, you can sign up as a special “bridge” relay that is not listed in any directory.
• No need to be an “exit” (so no abuse worries), and you can rate limit if needed
• Integrated into Vidalia (GUI)
• https://bridges.torproject.org/ will tell you a few based on time and your IP address
• Mail bridges@torproject.org from a gmail address and you’ll receive a few in response
Tails

• Privacy for anyone anywhere
• Linux live distro focused on Privacy
• Use the Internet anonymously and circumvent censorship
  • Tor network
• Leave no trace
  • No persistent data on the computer you are using unless you ask it explicitly
• Use state-of-the-art cryptographic tools
  • E.g., https everywhere addons
What We Have Learned

- Anonymization network
- Filtering vs. Censoring
- The Onion Router (TOR)
- Hidden Service (Dark web)
- Bridge Relays
Extra content on pentesting/firewalls
Policy Actions

• Packets flowing through a firewall can have one of three outcomes:
  – **Accepted**: permitted through the firewall
  – **Dropped**: not allowed through with no indication of failure
  – **Rejected**: not allowed through, accompanied by an attempt to inform the source that the packet was rejected

• Policies used by the firewall to handle packets are based on several properties of the packets being inspected, including the protocol used, such as:
  – TCP or UDP
  – the source and destination IP addresses
  – the source and destination ports
  – the application-level payload of the packet (e.g., whether it contains a virus).
Firewall Types

- **packet filters (stateless)**
  - If a packet matches the packet filter's set of rules, the packet filter will drop or accept it

- "*stateful*" filters
  - It maintains records of all connections passing through it and can determine if a packet is either the start of a new connection, a part of an existing connection, or is an invalid packet.

- **application layer**
  - It works like a *proxy* it can “understand” certain applications and protocols.
  - It may inspect the contents of the traffic, blocking what it views as inappropriate content (i.e. websites, viruses, vulnerabilities, ...)

4/25/23 CS166 L20
Stateless Firewalls

A stateless firewall doesn’t maintain any remembered context (or “state”) with respect to the packets it is processing. Instead, it treats each packet attempting to travel through it in isolation without considering packets that it has processed previously.

Allow outbound SYN packets, destination port=80
Allow inbound SYN-ACK packets, source port=80
Stateless Restrictions

- Stateless firewalls may have to be fairly restrictive in order to prevent most attacks.

---

**Diagram:**
- **Client** (trusted internal network)
- **Attacker**
- **Firewall**
  - Allow outbound SYN packets, destination port=80
  - Drop inbound SYN packets
  - Allow inbound SYN-ACK packets, source port=80

**Example:**
- SYN Seq = y
- Port=80
Stateful Firewalls

• **Stateful firewalls** can tell when packets are part of legitimate sessions originating within a trusted network.

• Stateful firewalls maintain tables containing information on each active connection, including the IP addresses, ports, and sequence numbers of packets.

• Using these tables, stateful firewalls can allow only inbound TCP packets that are in response to a connection initiated from within the internal network.
Linux Firewall

• **iptables** manage IP table rules
  – **Iptables**: `-L` to list active rules, `-A **chain**` to add rule
  – `-D **chain**` to delete rule, `-F` to flush rules

• **Stop ping**
  – `$ sudo iptables -A INPUT -p icmp --icmp-type echo-request -j **REJECT**`
  – `$ sudo iptables -A INPUT -p icmp --icmp-type echo-request -j **DROP**`
  – `$ sudo iptables -F`

• **For practicing:**
  – https://tryhackme.com/room/redteamfirewalls
What Is a Penetration Testing?

- Testing the security of systems and architectures from the point of view of an attacker (hacker, cracker ...)
- A “simulated attack” with a predetermined goal that has to be obtained within a fixed time
Authorization Letter

• Detailed agreements/scope
  – Anything off limits?
  – Hours of testing?
  – Social Engineering allowed?
  – War Dialing?
  – War Driving?
  – Denials of Service?
  – Define the end point

• Consult a lawyer before starting the test
Closed Box vs. Open Box

- It treats the system as a closed/opaque box, so it doesn't explicitly use knowledge of the internal structure.

- It allows one to peek inside the "box", and it focuses specifically on using internal knowledge of the software to guide the selection of test data.
Practical Techniques – Penetration Testing

1) Gather Information
2) Scan IP addresses
3) Fingerprinting
4) Identify vulnerable services
5) Exploit vulnerability (with care!)
6) Fix problems?
Fingerprinting

- What web server is running?
- What accounts have I found?
- What services are running?
- What OSes are running?
- Who is logged in?
- Is there available information on the web site?
Identify Vulnerable Services

• Given a specific IP address and port, try to gain access to the machine. Report all known vulnerabilities for this target.

• Nessus

• Nexpose
Vulnerability scanning

Nessus is the leader tool in vulnerability scanning

- There are two components:
  - `nessusd` server with ‘plugins’ list of known vulnerabilities (there are different kinds of subscription depending on how old the plugins are)
  - `nessus` is the front end of the tool. There are several version for windows and linux systems
Introduction to Nessus

- Created by Renaud Deraison
- Currently Maintained by Tenable Network Security
- Uses the NASL Scripting language for its plugins (currently over 13,000 plugins!)
- Price is still Free! But no more open source
- Register to obtain many NASL plugins (7 day delay).
- Or Purchase a Direct Feed for the Latest!
Nessus Features

• Client/Server Architecture
• SSL/PKI supported
• Smart Service Recognition
  – (i.e. FTP on 31337)
• Non-Destructive or Thorough Tests
• Vulnerability Mapping to CVE, Bugtraq, and others
• Vulnerability Scoring using CVSS from NIST.
The remote host is vulnerable to a denial of service attack in its SMB stack.

An attacker may exploit this flaw to crash the remote host remotely, without any kind of authentication.

Solution: http://www.microsoft.com/technet/security/bulletin/ms02-045.mspx
Risk factor: High
CVE: CVE-2002-0724
BID: 5556
Other references: OSVDB:2074
<table>
<thead>
<tr>
<th>Tool</th>
<th>UNIX</th>
<th>Windows</th>
<th>TCP scan</th>
<th>UDP scan</th>
<th>Host discovery</th>
<th>Port scanner</th>
<th>OS fingerprinting</th>
<th>DOS</th>
<th>Anonymity level</th>
</tr>
</thead>
<tbody>
<tr>
<td>SATAN</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>Medium</td>
</tr>
<tr>
<td>SARA</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>Medium</td>
</tr>
<tr>
<td>Nessus</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Medium</td>
</tr>
<tr>
<td>Advanced IP scanner</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>Medium</td>
</tr>
<tr>
<td>Advanced port scanner</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Medium</td>
</tr>
<tr>
<td>Strobe</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Medium</td>
</tr>
<tr>
<td>Udp_scan</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Low</td>
</tr>
<tr>
<td>Netcat</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>Low</td>
</tr>
<tr>
<td>Xprobe</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>Low</td>
</tr>
<tr>
<td>SoftPerfect Network Scanner</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Low</td>
</tr>
<tr>
<td>Angry IP Scanner</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Low</td>
</tr>
<tr>
<td>GFI LANGuard Network Scanner</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Low</td>
</tr>
<tr>
<td>Superscan</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Medium</td>
</tr>
<tr>
<td>ScanMinder: Standard</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Medium</td>
</tr>
</tbody>
</table>
Exploit vulnerability

• Try to exploit detected vulnerabilities, for example:
  – Buffer overflow
  – Heap overflow
  – SQL injection
  – Code injection
  – Cross-site scripting
• Metasploit is a framework that allows to test attacks
msf > use windows/smb/ms05_039_pnp
msf exploit(ms05_039_pnp) > set RHOST 192.168.10.110
RHOST => 192.168.10.110
msf exploit(ms05_039_pnp) > set TARGET 0
TARGET => 0
msf exploit(ms05_039_pnp) > set PAYLOAD windows/meterpreter/bind_tcp
PAYLOAD => windows/meterpreter/bind_tcp
msf exploit(ms05_039_pnp) > exploit
Pen Testing tools

• Often open source and a with a limited free version

• A good starting point is using a Linux Distro

• The most used distribution is Kali Linux – an open-source, Debian-based Linux distribution
  • Penetration Testing, Security Research, Computer Forensics
  • https://tools.kali.org

• When Things Get Tough…
Target machines

• You can find in the competitions like Capture The Flags

• In this tutorial we use Metasploitable 2 released by Rapid7
  – Rapid 7 manages Metasploit Framework

• Usually the target machines are in
Enumeration with Nmap (Network Mapper)

Port Division
- open, closed, filtered, unfiltered, open|filtered and closed|filtered

Scanning techniques
-\texttt{-sS} (TCP SYN scan)
-\texttt{-sT} (TCP connect() scan)
-\texttt{-sU} (UDP scans)
-\texttt{-sA} (TCP ACK scan)
-\texttt{-sW} (TCP Window scan)
-\texttt{-sM} (TCP Maimon scan)

\texttt{--scanflags} (Custom TCP scan)
-\texttt{-sI} <zombie host[:probeport]> (Idlescan)
-\texttt{-sO} (IP protocol scan)
-\texttt{-sN; -sF; -sX} (TCP Null, FIN, and Xmas scans)
-\texttt{-b <ftp relay host>} (FTP bounce scan)
Identify active hosts and services in the network

• **ping sweep** useful to identify targets and to verify also rogue hosts
  
  Ex:
  
  – `nmap -v -sP 10.0.2.0/28`
    * -sP Ping scan.

• **port scanning** useful to identify active ports (services or daemons) that are running on the targets
  
  Ex:
  
  – `nmap -v -sT 10.0.2.x`
    * -sT normal scan
    * -sS stealth scan –sV services